Project 2 Personalized Recommendations

1. Problem Description

In previous lessons, we learned that personalized recommendation is a typical application of big data. Personalized recommendation is to use the known user browsing history to recommend new items: given a user behavior matrix $X_{m\times n}$, where m and n are the number of users and items, and $X_{i,j}$ is the user i's rating towards an item j. Therefore, the so-called recommendation task translates into how to predict unknown values in the matrix when we know part of them. In this project, the task is to complete the movie recommendation task from Netflix.

2. Dataset

We're using a subset of the Netflix recommendation contest, with 10,000 users and 10,000 movies. The user behavior data contains the user's rating of the video, and the score is valued from 1-5. We select 80% of the behavioral data as the training set and the remaining 20% as the test set.
We have downloaded the data and provide a **data.zip** file for you, which includes:

(1) The user list: *users.txt*
The file has 10000 lines, one integer each line, representing the user's id, and the file contains all users of this Project.

(2) Movie name: *movie_titles.txt*
The file contains the year and name of each movie in the following format: "movie id, year, movie name". The three terms are separated by commas in the txt file.
Note: Information such as rating time and movie titles in the dataset will only be used for the optional homework part.

(3) Training set: *netflix_train.txt*
The file contains 6.89 million user ratings, each line represents one piece of rating, in the form of: "User id, Movie id, Rating, Rating Date", Where the user id appears in the user .txt, the movie id is an integer of 1 to 10000. The four terms are separated by space in the txt file.

(4) Test set: *netflix_test.txt*
The file contains approximately 1.72 million user ratings in the same format as the training set.

3. Experiment

(1) Data pre-processing.

The first step you have to go through during the big data processing is data cleaning and formatting. Back to our problem, the data has been cleaned, so just organize the input file into a matrix X of dimensions $[\text{user number}, \text{movie number}]$, and $\text{the element of the matrix } \mathrm{X_{ij}}$ corresponds to the user i's rating towards the movie j. For movies with unknown ratings, there are some specific ways to deal with them, such as setting them all to 0 or building another matrix to record what is

known and what is unknown.

Considering the computing performance of different computers, if you find it difficult to process a matrix of 10000 x 10000 with your PC, you can sample a subset of users and some movies and explain the sample rules in the lab report (the subset size should be no less than 2000 x 2000).

Note: It has been tested that even when using full data, the required memory can be controlled at no more than 4GB (provided that the data, algorithms are processed properly).
The output of this step is two matrices, respectively, the training set X_{train} and the test set $X_{test.}$

(2) Collaborative filtering

Collaborative filtering (CF) is one of the most classical recommendation algorithms, containing the user-based CF and item-based CF. This time, we should implement the user-based CF algorithm. The idea of the algorithm is simple, when we need to predict whether user i will like movie j, we just refer to the users that are similar to user i and check their ratings towards movie j, and then average their ratings based on similarity coefficients. Expressed in formulas, that is:

$$\text{score}(i,j) = \frac{\sum_k sim\big(X(i), X(k)\big) score(k,j)}{\sum_k sim\big(X(i), X(k)\big)}$$

Where X(i) is the ratings of user_i towards all movies, and in our problem X(i) is the i^{th} row of the matrix X_{train} (a vector of dimension 10000 with unknown elements set to 0).
$sim\big(X(i), X(k)\big)$ Represents the similarity of user_i and user_k based on the ratings towards the movies, can be calculated by the cosine similarity of the two vectors:

$$cos(x,y) = \frac{x \cdot y}{|x| \cdot |y|}$$

With the above formula, we can calculate the user's possible rating for each record in the test.
Note: In the above formula, we do not consider the range of k. You need to give a reasonable strategy to choose a proper range for k to give better prediction.
In this project, we adopt the RMSE (Rooted Mean Square Error) as the evaluation metric as follows,

$$\text{RMSE} = \sqrt{\frac{1}{|Test|}\left(\sum_{\langle i,j\rangle \in Test} \left(X_{ij} - \tilde{X}_{ij}\right)^2\right)},$$

where Test is the set of all test samples and |Test| is the size of the set (number of testing samples), $\tilde{X}_{ij}$ is the prediction value and X_{ij} is the ground truth value.

(3) Matrix decomposition algorithm based on gradient descend

In the previous courses, we described the application of matrix decomposition in the recommendation algorithm. For a given behavioral matrix X, we decompose it into the product of two matrices U, V, so that the product of U and V approximates the known value part, i.e. $X_{m*n} \approx U_{m*k} V_{n*k}^T$, where k is the latent space dimension, which is the hyper parameter of the algorithm. Based on the low rank hypothesis of the behavioral matrix, we can regard U and V as the representation of the user and the movie in latent space, and their product matrix can be used to predict unknown parts.
In this experiment, we use gradient descent to optimize to solve this problem. The target function of our recommendation algorithm is:

$$J = \frac{1}{2}\|A \odot (X - UV^T)\|_{\text{F}}^2 + \lambda\|U\|_F^2 + \lambda\|V\|_F^2$$

Where A is the indicator matrix, $A_{ij} = 1$ means the value X_{ij} is known and vice versa. $\odot$ is the Hadama product (i.e. the matrix is multiplied by elements). $\|\cdot\|_F$ is the Frobenius norm of a matrix, calculated as $\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$. In the target function, the first item is the known value part, the error between the approximation and X. The latter two are regularization terms that are added to prevent overfitting, and λ is the parameter that controls the regularization terms defined by yourselves.

When the target function gets the minimum value, the algorithm will get the optimal solution. First, we calculate the partial gradient of U and V as follows:

$$\frac{\partial J}{\partial U} = \big(A \odot (UV^T - X)\big)V + 2\lambda U$$

$$\frac{\partial J}{\partial V} = \big(A \odot (UV^T - X)\big)^T U + 2\lambda V$$

After that, we iterate update U and V with gradient descent, as follows:

Initialize U and V

Loop until convergence

Calculate $\frac{\partial J}{\partial U}, \frac{\partial J}{\partial V}$

Update $U = U - \alpha \frac{\partial J}{\partial U}, V = V - \alpha \frac{\partial J}{\partial V}$

End loop

The α in the algorithm is the learning rate, usually a smaller number is selected on a case-by-case basis. In this job, you can try real values from 1e-5 to 1e-1. To judge whether the algorithm converges or not, you can use the target function, if its change between iterations is less than a certain threshold, it means this algorithm has converged. Or you can also set a maximum number of iterations J to end this algorithm.

(The above is the most classic batch gradient descent, you can also learn and try some other optimizers, such as SGD, Adam, etc.)

To this end, we have completed the process of optimizing the target function under the given k and λ. You can draw the target function value of each iteration and the RMSE value on the test set to observe the changes of these two metrics until the algorithm converges. In addition, you need to try different k and λ values to find the best parameters for the algorithm.

4. Report submission requirements

You only need to submit reports and codes, please do not submit **intermediate results and data files.** The report needs to include the following:

(1) A simple description of the data preprocessing phase and whether the full amount of data was used at the end. **(20 points, 1-5 points discount without using full data).**

(2) User-based collaborative filtering implementation. Contains a brief introduction to the code, the final RMSE, and the time consumption of the algorithm. **(20 points).**
Note that because the X matrix is very sparse in this experiment, use matrix operations whenever possible when using tools such as matlab that are particularly optimized for sparse matrices, and avoid using for loops. How our formula is translated into a matrix form of operation, please derive on your own.

(3) The result of the matrix decomposition algorithm. **(40 points)** The report needs to include:
   a) For a given case of k=50, λ=0.01, draw the changes of the target function value and RMSE on the test set during the iteration, give the final RMSE, and make a simple analysis of the results.

b) Adjust the values of k (e.g. 10,50,100) and λ (e.g. 0.001,0.01,0.1, 1), compare their final RMSE, and select the optimal combination of parameters.

(4) Compare the results of (2) (3) and discuss the advantages and disadvantages of the two approaches. **(20 points).**

5. The optional content

This Project contains a 5-point optional part for students interested in recommendation problems for in-depth study: In this dataset, we provide additional data such as the date of the user's rating behavior, the movie name, and more. These data provide an additional information for us to assess the similarity of scoring behavior and the similarity of movies. How can this information be used to improve our recommendation algorithms and further improve results?

Students interested in the optional content, directly put forward the method and write the corresponding experimental results into the experimental report.

Note: If only a general idea is provided, such as "consider the year information of different films" without specific algorithmic and experimental support, no bonus points will be obtained.

6. Submission request:

You should submit one compressed file with:

(1) One report in English. It should be no more than 8 pages.

(2) If you have codes (e.g. Python, Matlab, Java, etc.), put them into one file. Do not submit multiples files for codes. You don't need to submit codes if you use SPSS/excel/etc.

(3) Do not submit other files, e.g., intermediate results. Figures should be included in the report rather than in separate files.

(4) All files should be named with your ID, e.g., 2020001002.pdf, 2020001002.py, 2020001002.zip. Please check carefully (because we need to put your files into a system to check for plagiarism).

(5) Failing to satisfy the above requirements will cost 5% of your grades.

(6) Late submissions will cost 20% of your grades per week. The number of weeks is calculated by rounding up, e.g., if your submission is late by 1 day, it will be accounted for as $\lceil 1/7 \rceil = 1$ week. **Start early!**

(7) Each student should do his/her homework independently. Do not copy codes, reports, results, or any material from others, or share them with others, including online and publicly available sources, *e.g.* in GitHub. One exception is to use any build-in function in the software, e.g., in Excel, Python, MATLAB. Note that both copying from others and intentionally providing materials for others to copy are considered plagiarism (so do not send your codes, results, reports to others) and will result in 0 grades and/or failing the class and/or other consequences instructed by the school honor code. It is a "red line" and we take it seriously

(8) Note the way you use the provided data. You could only use the data for the homework and should not distribute the data for any other purpose on any public platform, or you will be liable for the accident.

7. Some tips:

(1) Judge the reasonable range of RMSE values based on the actual background of this task and the physical meaning of RMSE. For example, if we use random guess or all 3 points as the baseline method, what is its RMSE? This will help you determine whether your program is executing correctly.

(2) Note that the values that do not appear in the matrix represent a movie that the user has not seen, and instead of the user rating the movie 0. This is also why we add an indication matrix A to do hadama product during matrix decomposition.

(3) In matrix decomposition algorithms, the initialization of U and V also has an impact. What is the physical meaning of U/V in the actual context of this assignment? What range should their initialization be in?